Implementing And Debugging Cheatsheet

Before Implementing:

1) Write down the exact steps of your algorithm
2) Write down what all the complicated variables do

While Implementing:

3) Get a non-trivial test case of size N = 6 or N = 7
    a) This could be the sample input or a test that you make
4) After implementing each step defined in the before implementing section, print everything to ensure everything is correct

Debugging:

1) Calculate the probability that if you create 7-15 small test cases, one of them will fail based on which tests you are currently getting correct and wrong
    a) If this probability is tiny, then don't waste time trying to debug. You are like getting most of the tests anyways if the probability is tiny.
2) Find a test case your code fails on
    a) Create 7-15 test cases of size N = 6 or N = 7. Solve these test cases by hand and see if the answer matches what your code output.
3) Use the test case that your code fails on to find the location of your bug
    a) Binary search on the location of the bug.

Implementation Tips (to avoid integer overflow):

- Make all the variables longs
    - In Java, the array indexes should be ints
    - In C++:
      ```
      #define int long long
      …
      main () {
      ```
- If you have to give the answer mod something, create functions for add, subtract, and multiply. For example:

    -
      ```
      long add (long a, long b) {
              return (a+b) % MOD;
      }
      ```

Common Bugs:

- Failing the sample input on USACO
    - Binary search on the location of the bug
- USACO gives the verdict of !
    - This is either memory limit exceeded or array out of bounds. See the video on checking if you are using too much memory to figure out which of these two potential reasons is causing this error.
    - If it is array out of bounds, binary search on the location of the bug
- First few tests are correct, last couple are ‘x’
    - Integer overflow. Implement the tips to avoid overflow above to fix this
- First few tests are correct, last couple are !
    - Memory limit exceeded. This likely comes from a data structure that starts empty and has values added to it like a priority queue.